The Evaluation of Just-In-Time Hypermedia Engine

Zong Chen[1], Li Zhang[2]
[1](School of Computer Sciences and Engineering, Fairleigh Dickinson University, USA)
[2](Computer Science Department, New Jersey Institute of Technology, USA)

ABSTRACT: *Just-in-time Hypermedia Engine generates documents and displays screens in response to user queries "dynamically". This paper implements and evaluates the JHE engine performance.*

Keywords - *Just-in-time, Hypermedia*

1. INTRODUCTION

Many analytical applications, especially legacy systems, create documents and display screens in response to user queries "dynamically" or in "real time". These documents and displays do not exist in advance, and thus hypermedia must be generated "just in time"—automatically and dynamically.

This paper implements a Just-in-time Hypermedia Engine based on the design proposed in [1]. Figure 1 shows the JHE architecture.

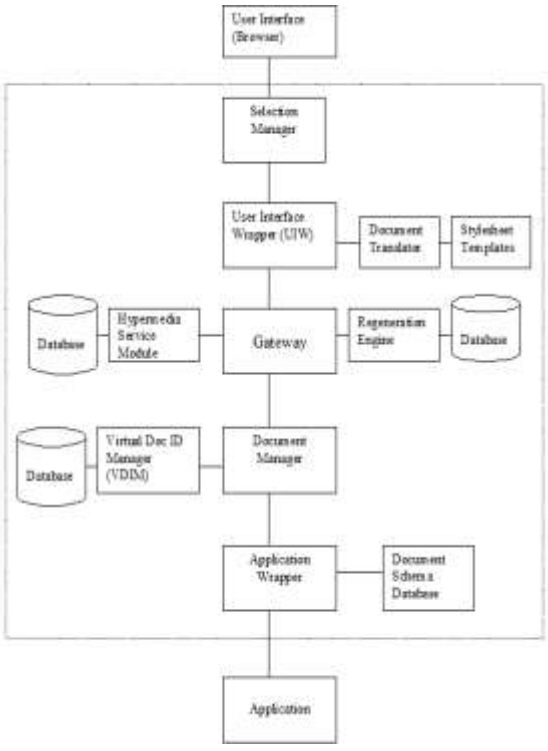

Figure 1. JHE Architecture.

JHE is a Web server that can integrate many external applications and supply hypermedia functionality for them. JHE uses Apache Tomcat Web server (http://jakarta.apache.org/tomcat/) and Java language to do programming. Most JHE modules are programmed in Java to run together with the Apache package. JHE uses JavaScript language to program on user interface design, which is part of the User Interface Module. The Selection Manager also uses some JavaScript functions to get selections from window.

The User Interface Wrapper (UIW) contains several User Interface (UI) menus to allow users to add bookmarks, comments and links to a virtual document. The Selection Manger (SM) generates anchor information. The Document Manager (DM) finds all anchors related to a virtual document, and find the exact positions of the anchors, and re-identify the anchor content. The Hypermedia Service Module (HSM) stores and retrieves bookmark, comment and link information into database. The Regeneration Engine (RE) revalidates the document. The Application Wrapper (AW) intercepts the application document and parses the document into an XML document.

2. JHE Implementation

2.1 User Interface Wrapper

The user interface is composed of four parts: application list, main window, menu list, and menu window as in Figure 2. The upper left screen shows the application list that JHE supports. Currently JHE only supports the NSSDC application, but will support more applications in the future. The upper right screen shows the virtual document from applications. The lower left screen shows the hypermedia functionality that JHE supports, this include bookmark list, add new bookmark, add new comment, and add new link menu. The lower right screen shows the menu content, which allows users to enter some information or select some information from menu. The UIW contains Bookmark Service Menu, Link Service Menu and Comment Service Menu. Each service menu is an HTML page containing some input parameters and JavaScript functions.

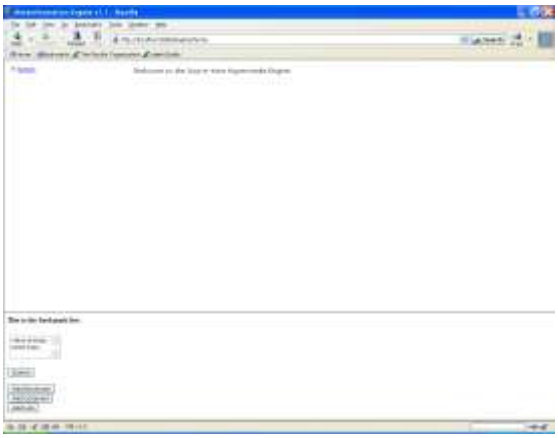

Figure 2. JHE Main Menu.

When a user wants to add a bookmark, he clicks the "add bookmark" button on the screen, then the bookmark service menu will appear at the right corner of the window. The Bookmark Service Menu is an HTML page with a "form" to allow users to input parameters from screen, such as the bookmark name and revalidation criteria.

When a user wants to add a link, he clicks the "add link" button on the screen, then the link service menu will appear at the right corner of the window. The Link Service Menu is an HTML page with a "form" to allow users to input parameters from screen, such as the link name and link destination. When the user clicks the "submit" button, UIW packs the information and sends it to Gateway. A link destination could be a URL entered by the user, an anchor or a bookmark. When UIW composes the Link Service Menu, it retrieves all anchors and bookmarks and lists them in the menu. If a user wants to add a link to a bookmark or an anchor that has not existed, he should create it first and then makes a link to it.

When a user wants to add a comment, he clicks the "add comment" button on the screen, and then the comment service menu will appear at the right corner of the window. The Comment Service Menu is an HTML page with a "form" to allow users to input parameters from screen, such as the comment name and content.

2.2 Document Translator

The Document Translator transforms the XML document into a displayable HTML format. In JHE, the HTML file also includes displayable parts (source document from application) and non-displayable parts. The non-displayable parts include system data (e.g. document identifiers and bookmark information list) and script functions. The script functions implement some of user interface functions. The Document Translator is a Java class running on the JHE server.

2.3 Selection Manager

When a user selects some texts from screen, then clicks a button, the Selection Manager will generate an XPointer expression and then sends it to the Gateway. The Selection Manager is implemented by Mozilla's XPointer Lib (http://xpointerlib.mozdev.org/).

2.4 Hypermedia Service Module

The Hypermedia Service Module stores and retrieves hypermedia construct information into the JHE database. There are three hypermedia functionality menus displayed on the UI. They are the Bookmark Menu, Comment Menu and Link Menu. These Menus are parts of the UIW. They are HTML pages that allow users to enter parameters and submit the information to the Gateway. The Gateway forwards the hypermedia functionality command and parameters to the HSM. HSM parses the command and stores the hypermedia construct information into the JHE database. When the user clicks the hypermedia construct icons on the screen, the HSM receives the hypermedia functionality command and parameters from the Gateway. The HSM retrieves the hypermedia construct information from the JHE database and then sends it to the Gateway. The UIW receives the information from Gateway and displays the information on the UI.

2.5 Regeneration Engine

The Regeneration Engine performs the following steps to regenerate a virtual document: (1) get query command and parameters from database by the unique document identifier; (2) send a request to the AW and then get document back;(3) validate the virtual document by different criteria.

RE receives the source document, then revalidates for different criteria.

Criteria 0: Exact copy. This means it needs to do byte comparison between the new document and the stored document.

Criteria 1: Values can change. This means it needs to compare the document structure.

Criteria 2: All related queries. This means do not need to compare with history information.

When the user creates a bookmark, the "content" value in the bookmark record is the file name. If the "Criteria" is "0", RE retrieves the file name from database, and reads the original document from the specific directory that stores documents. Then it compares the original document with the new document byte by byte. If the "Criteria" is "1", RE calls the AW to parses the regenerated document according to a stored document template. If the "Criteria" is "2", then RE does not compare anything.

If revalidation is successful, then the Document

Manager will do relocation and re-identification for all related anchors for the virtual documents, otherwise it will give warning messages to users.

2.6 Document Manager

The Document Manager gets all related anchors from the JHE database, then relocates and re-identifies anchors according to different criteria. The Document Manager performs the following steps:

(1) Get an anchor list from JHE database. This includes all global anchors and other local or specific anchors that have the same document identifier with this virtual document.

(2) Evaluate each of these anchors according to the criteria. First, the Document Manager finds the anchor location according to document structure. Then, it compares the element value with the history information if the criterion does not allow element value to change; it does not compare the element value if the criterion allows element value to change. The Document Manager uses Apache's XPath tool (which is a Java class package that evaluates XPath expressions in a JDOM document) to find the anchor locations.

(3) The Document Manager finds all related hypermedia constructs for each anchor (comment, links, etc.), then it attach hypermedia objects to this document by inserting icons at the side of the elements.

2.7 Application Wrapper

The Application Wrapper (AW) intercepts the application commands and catches the source documents from applications. Then it parses the source document and translates it into a well-structured XML document.

When a user clicks the button, the UIW will send the command "/jhe/sc" together with the parameters to the JHE Gateway, and then the Gateway forwards it to the destination Application Wrapper (AW). The AW parses the query result, maps the JHE destination and command to the real application destination and command. Then the AW sends an HTTP request to the destination application with parameters (the real URL of the application). The real application executes the command and parameters and sends back the document to the AW. After the AW parses the source document and gets elements' values, the document is restructured and translated into an XML document based on the document structure. While the current instantiation of a document is parsed automatically, usually the document structure is analyzed manually in advance.

3. JHE EVALUATION

In this paper, NASA's National Space Science Data Center (NSSDC) Web system is used to integrate with JHE for evaluation purpose. T96 model is one of NSSDC models for earth magnetic field calculation.

3.1 Dynamic Regeneration

When user selects the "T96 Model" command (which is read from the system setting file when JHE starts up) from the UI menu, it gives users the T96 homepage, which has the query table. This allows users to enter in parameters. Before the user creates a new bookmark, he needs to enter and submit parameters to JHE to create a new calculation result. After he adds a new bookmark for the result, he will not need to enter parameters again. JHE will do regeneration automatically.

Figure 3 shows the T96 application module integrated into JHE. Figure 4 shows the calculation result after the user enters some parameters. When the user selects the "add bookmark" command from the menu at the right corner of the window, the Bookmark Service Menu will prompt the user to enter in bookmark information, such as name ("exact copy" in the example) and criteria ("Only this query result"). Then after the user clicks "submit", a new bookmark named "exact copy" is added into database.

When the user revisits the bookmark, he can select the bookmark from the list (in left corner of the window). The JHE would then regenerate the result without asking the user to reenter parameters. Figure 5 shows the regenerated document and the message shows the regeneration is successful.

Figure 3. T96 home page integrated into JHE

Figure 4. Add a bookmark.

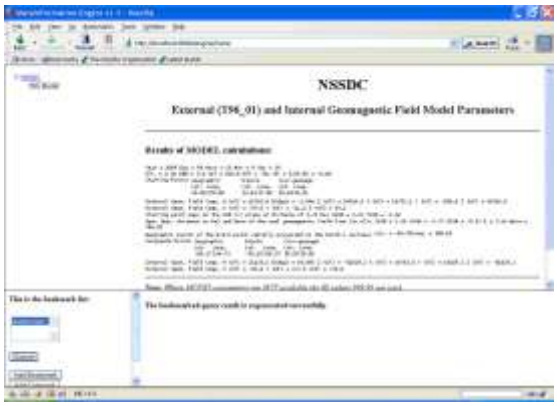

Figure 5. Regenerated document with criterion 0.

Compare the JHE generated virtual documents with the original virtual documents generated by the application. JHE integrates applications into the hypermedia system and it intercepts the source documents from applications. Since the original virtual document is translated into a well-structured document, it is important that data from application be the same as the original. For example, the calculation result should be exactly the same as the original although the format may be a little different. Also, users are expecting to get a similar layout document, such as font size, line width, and background. JHE translates the original document into an XML document, and then generates some style file for this document to have a similar layout as the original.

When doing regeneration, there is a question whether the re-generated document is the same as the original. Depending on the dynamics of the application systems, a virtual document could change frequently or rarely. The revalidation process gives users some information about the comparisons between the newly generated document and the original. The three levels of revalidation give users flexibility to perform hypermedia functionality on a virtual document. If the revalidation is not successful, this does not mean the virtual document is wrong. It just gives some warning message to the user that the newly-generated virtual document has changed according to the user's criterion. It may remind the user that the data is out-of-date, and he may need some changes. Unlike a traditional bookmark, which just remembers the address of the document, it does not have any history information about it. By doing this kind of revalidation, it can help the user to compare the regenerated virtual document with the original and remind him to do some changes, which is really better than traditional bookmark.

3.2 User Declared Comment

User can create a comment by following steps: (1) click the "add comment" button; (2) select some texts from screen; (3) specify granularity; (4) edit comment content; (5) edit comment title; (6) click "submit" button.

After the user clicks "submit" button, JHE stores comment information into database. When the user revisits the virtual document, JHE finds marked anchors in the document and re-identifies the anchors according to different criteria, and then inserts an icon at the side of the element. In Figure 6, the anchor "year=2005" is relocated and re-identified after regeneration. When the user clicks on the (H) icon, it pops up a window shows a list of hypermedia constructs, such as comments and links, show in Figure 7 (there are two comments and one link). When the user clicks on the comment, the window will display corresponding comment contents, shown in Figure 8.

Figure 6. Regenerated virtual document with hypermedia constructs attached.

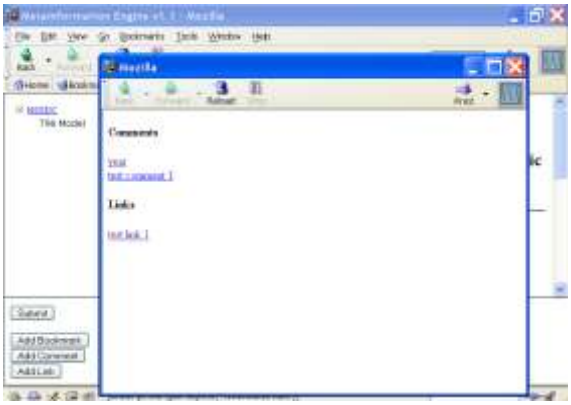

Figure 7. List of comments and links.

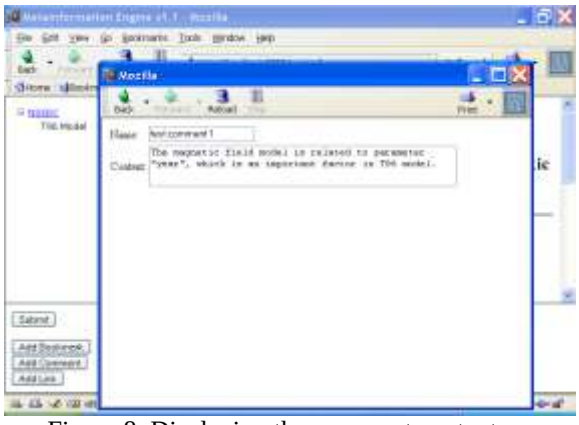

Figure 8. Displaying the comment contents.

3.3 Manual Link

A user selects some texts from screen then enters parameters in the Link Service Menu and submits the request to JHE. JHE adds a link into database. Next time the user clicks the link title in the hypermedia constructs window to traverse a link.

Creating a manual link has following steps: (1) click the "add link" button; (2) select some texts from screen; (3) select destination; (4) specify granularity; (5) edit link title; (6) click "submit" button.

There are three types of link destinations: a URL, an anchor, and a virtual document in JHE. An anchor is a selected anchor in a JHE supported virtual document. A virtual document is a JHE supported virtual document that can be generated by JHE integrated applications. Depending on different types of link destination, the link traversal is different. For a link that has a specific URL, clicking on the link will take the user to another Web page. For an anchor, clicking on the link will take the user to the virtual document which JHE regenerates, and the destination anchor is highlighted. For a virtual document, clicking on the link will take the user to the virtual document which requires JHE to do regeneration. Also, for the destination document, hypermedia objects are attached on it.

Figure 9. Add a link whose destination is a bookmark.

3.4 Relocation and Re-identification

Relocation means to find the location of the anchor when the byte offset of the anchor has changed. The anchor menu is part of the Comment Service Menu (when the user adds a comment) or the Link Service Menu (when the user adds a link). The anchor menu allows users to specify anchor granularity and re-identification criteria. Relocation is based on the anchor location and the granularity.

There are three types of anchor granularity: global, local and specific. "Global" means the anchor can appear in any document that has the same element. "Local" means the anchor can appear in the same element anywhere in a particular document. "Specific" means the anchor can only appear at a particular location in a particular document.

Figure 10 shows how to create a "local" anchor on the selection "degree" in the document entitled "value change" when the user adds a comment on the anchor. The "degree" element is a sub element under element "Internal Geom. Field". It appears twice in this document. Figure 11 shows revisiting the document entitled "value change" by clicking on the "value change" bookmark. Since it is a "local" anchor, all "degree" elements in this document are attached with hypermedia constructs. By clicking on the icons besides same elements, users can see the same comment content.

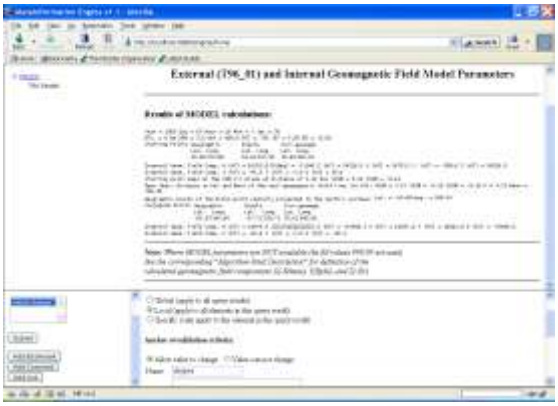

Figure 10. Create a "local" anchor.

Figure 11. Relocate "local" anchors.

After finding the anchor's location, JHE needs to

identify that the relocated anchor is the same one as the originally marked one. There are two criteria for re-identification: element's value can change; element's value can not change.

Figure 12 shows an anchor's re-identification criteria is "value can not change". Figure 13 shows revisiting the document. Since the original value is "2", the new value is "0", they are not same. JHE gives a warning message by adding a pink icon at the side of the element. For those valid elements, the icons attached are blue.

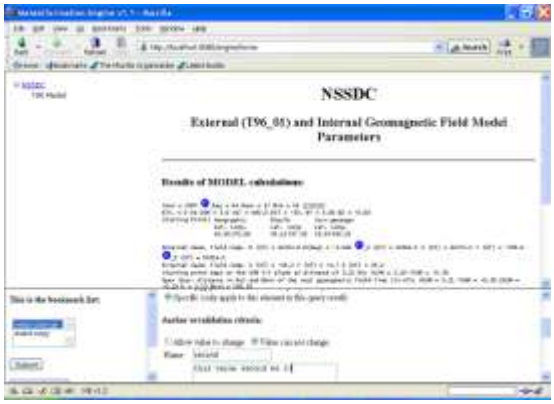

Figure 12. Add a "no value change" anchor.

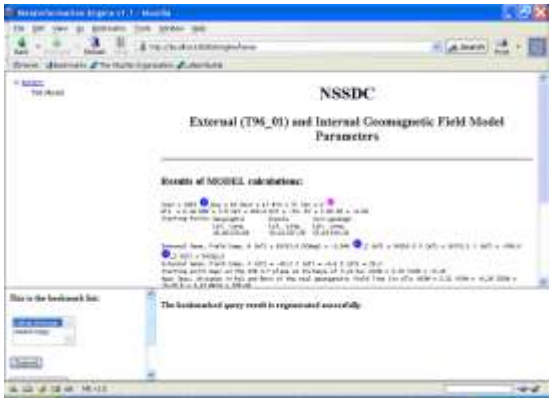

Figure 13. Revalidate anchors.

4. Conclusion

Hypermedia engines differ in how applications integrate with hypermedia systems; what kind of hypermedia functionality the hypermedia engine can supply, and how applications communicate with the hypermedia engine. Notable projects include Microcosm's Universal Viewer [2], Freckles [3], the OO-Navigator [4], WebVise [5], SFX [6], InfiniTe [7], and Dynamic Hypermedia Engine (DHE) [8]. Microcosm's Universal Viewer handles all communications without modification to application's source codes; WebVise and DHE use application wrappers to integrate with hypermedia systems. Early hypermedia engines (Microcosm and Freckles) only support first-generation hypermedia functionality (such as links and nodes); while WebVise and InfiniTe supports more hypermedia functionality and are Web-based. Many hypermedia engines communicate with applications by message passing using existing protocols (e.g, WebVise uses OHP); some use operating system specific tools to handle communications (such as Freckles uses window event manager).

The above hypermedia engines have many characteristics in common, such as integration with viewers, integration with information systems, support for identifiers of objects, internal document addressing and hypermedia functionality. None of them supports virtual documents, except DHE and SFX. DHE and SFX can analyze the underlying relationships based on the application information, then parse the source documents and insert links to these documents. Links are dynamic and automatically generated. However, they do not support regeneration, re-location and re-identification.

The Just-in-time Hypermedia Engine (JHE) is implemented in this paper. The Just-in-time Hypermedia Engine (JHE) executes as a middleware between an application and its user interface, providing additional hypermedia navigational, structural and annotation functionality, with minimal modification to the application.

The future work may include: (1) implementing more hypermedia functionality for the JHE system such as automatic linking and guided tours, (2) supporting virtual documents with binary formats such as images in web pages.

References

[1] Z. Chen, L. Zhang, "The Just-In-Time Hypermedia Engine", SSRG International Journal of Computer Science and Engineering, volume1 issue10: 50-55, 2014.

[2] H. C. Davis, S. Knight, and W. Hall, "Light Hypermedia Link Services: A Study of Third Party Application Integration." Proceedings of the 1994 European Conference on Hypermedia Technology, 1994

[3] C. Kacmar, "A Process Approach for Providing Hypermedia Services to Existing Non-hypermedia Applications", Journal of Electronic Publishing: Organization, Dissemination and Design. Vol. 8(1), 31–48, 1995

[4] A. Garrido, G. Rossi. "A Framework for Extending Object-Oriented Applications with Hypermedia Functionality." *The New Review of Hypermedia and Multimedia.* Vol.2: 25-41, 1996.

[5] K. Grønbæk, L. Sloth, and P. Qrbek. "WebVise: Browser and Proxy Support for Open Hypermedia Structuring Mechanisms on the WWW." Proceedings of the 8th International World Wide Web Conference: 232-267, 1999.

[6] H. Van de Sompel, and P. Hochstenbach. "Reference Linking in a Hybrid Library Environment, Part 2: SFX, a Generic Linking Solution." *D-Lib Magazine.* April, 1999

[7] K.M. Anderson, R.M. Taylor, and Jr. E.J. Whitehead. "Chimera: Hypertext for Heterogeneous Software Environments." Proceedings of the ACM Conference on hypertext: 94-107, 1994.

[8] R. Galnares, "Augmenting Applications with Hypermedia Functionality and Metainformation." Ph.D. Dissertation, New Jersey Institute of Technology, 2001.